Software Architectures

Assignment 1: Akka Streams

TA: Camilo Velázquez-Rodríguez
camilo.ernesto.velazquez.rodriguez@vub.be
F.10.725

Assignment

You will have to implement and report on your original solution to a problem using Akka Streams. You will find the input file needed for this assignment in *Assignments > Assignment 1.*

Deadline: January 5th 2025 by 23:59. The deadline is fixed and will not be extended for any reason. Submissions after the deadline will not be considered for grading and the grade will automatically be **ABS**.

Deliverables A report and source code. The report (in English) explains your solution to the problem and the main components used in your implementation. Please use simple diagrams to illustrate the architecture of your solution.

The report should be handed in as a single PDF file.
The file should follow the naming schema `FirstName-LastName-SA1.pdf`, for example: `Camilo-Velazquez-SA1.pdf`.

The source code is your project implementation zipped in a single file. Do not include files or folders resulting from the compilation process of your project, e.g., target/ or *.class. Submit the *report* and *source code* as a single zip file to the Software Architectures course page on Canvas, by clicking on *Assignments > Assignment 1.*

Plagiarism Any suspicion of plagiarism will be reported to the Dean of Faculty without delay. Both user and provider of such code will be reported and will be dealt with according to the plagiarism rules of the examination regulations (cf., OER, Article 118). The Dean may decide on (a combination of) the disciplinary sanctions, ranging from 0/20 on the paper of the given programme unit to a prohibition from (re-)enrolling for one or multiple academic years (cf., OER, Article 118$5).

Grading Your solution will be graded and can become the subject of an additional defence upon the lecturer's request.

Problem Description

In this assignment you will implement a Pipe and Filter architecture for a system that processes records of games of the National Collegiate Athletic Association (NCAA)[1].

You are provided with a **CSV** file named *basketball.csv* containing the data you need in this assignment. Within the file are 2,117 records of information related to the NCAA basketball games from the 1984-1895 to the 2017-2018 seasons. Each record contains 27 columns separated by a comma naming the attributes of this dataset. For example, you can find `season`, `round`, `game_date`, etc., as attributes in the dataset.

The task for this assignment is to extract information provided by the dataset using the Akka Streams library. Specifically, the implementation of the pipeline should answer the following questions:

- What is the number of victories of each team on a Sunday?
- What is the number of victories of each team with a difference of five or fewer points?
- How many times a team has reached at least the quarter-final (4) instances?
- How many losses have had the teams in the dataset between the years 1980 and 1990?

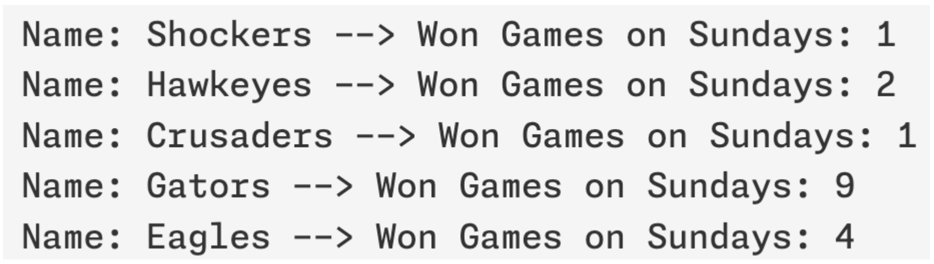
Given the mentioned **CSV** file you will have to create a **Source** from a hard-coded file path. In a follow-up step the Source with the **CSV** file should be parsed and subsequently processed to obtain a stream of `Maps` with the file information. Each map element should be transformed into an object instantiation of a user-defined class.

The stream of objects should be grouped by certain criteria to continue with the processing. You could group elements in the stream by any of the attributes in the dataset, e.g., `season`, `round`, `winning team`, etc. This will allow you to group records under the same defined criterion to answer the above questions. Later, a buffer will restrict the stream with a capacity of 20 elements and a *backpressure* strategy in case of overflow.

For each question, a custom **Flow** shape should be created. The shape should have a **balanced** approach to send groups of elements (depending on your previous grouping strategy) to two pipelines. The pipelines should work in the same way by counting the number of elements that satisfy a criterion (e.g., victories on Sundays). You should create the necessary classes to compute and store filtered elements. Both pipelines will be **merged** in a single outlet as the output of the custom shape.

Finally, the answer to each formulated question should be written to an external file. These files could be either in **TXT** or **CSV** formats. The information of each file should answer the questions. The screenshot in Figure 1 provides an example of the expected output for one of the files.

[1]`https://console.cloud.google.com/marketplace/product/ncaa-bb-public/ncaa-basketball`



```
Name: Shockers --> Won Games on Sundays: 1
Name: Hawkeyes --> Won Games on Sundays: 2
Name: Crusaders --> Won Games on Sundays: 1
Name: Gators --> Won Games on Sundays: 9
Name: Eagles --> Won Games on Sundays: 4
```

Figure 1: Screenshot of a fragment from the expected output.

In the above screenshot, a line consists of the name of a team and the number of games won on Sundays for this dataset. As such, each file will correspond to the answer of a formulated question. The above requirements should be implemented using Akka Streams and Scala 3. You are also required to use the `GraphDSL` to implement the custom **Flow**s. **Motivate your decisions** in the report as per the problem description and illustrate your approach using concepts from Akka Streams.

More information on Akka Streams and external libraries

- `https://doc.akka.io/docs/akka/current/stream/index.html`
- `https://doc.akka.io/docs/akka/current/stream/stream-graphs.html`
- `https://doc.akka.io/docs/alpakka/current/index.html`